ECGR 6118
Computer Project: Transforms
Student Name:_____

Review of transforms

dummy fnction x(t)

$x(t) := 1$

Continuous-time, 1D Fourier transform:

The 1-D continuous-time Fourier transform is:

$$X(\Omega) := \int_{-\infty}^{\infty} x(t) \cdot \left(e^{-i \cdot \Omega \cdot t}\right) dt$$

$$x(t) := \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} X(\Omega) \cdot \left(e^{i \cdot \Omega \cdot t}\right) d\Omega$$

Discrete Time Fourier Transforms (DTFT)

The DTFT (Discrete Time Fourier transform) is defined for the discrete time sequence x[n] as $X(\omega)$ given below

$$X(\omega) := \sum_{n=-\infty}^{\infty} \left(x(n) \cdot e^{-i \cdot \omega \cdot n}\right)$$

Recall that the DTFT is periodic, with period = 2π .

$$x(n) := \frac{1}{2\pi} \cdot \int_{-\pi}^{\pi} X(\omega) \cdot e^{i \cdot \omega \cdot n} d\omega$$

Discrete-time Convolution
(Linear Convolution, not Circular Convolution)

dummy fnction h(t)

$$h(t) := 1$$
$$T(t) := 1$$

Convolution for a discrete time sequence y[n] = x[n] * h[n] = h[n] * x[n]is given below

$$y(n) := \sum_{p=-\infty}^{\infty} (x(p)\cdot h(n-p))$$

or

$$y(n) := \sum_{p=-\infty}^{\infty} (h(p)\cdot x(n-p))$$

LTI (Linear Time Invariant) and Convolution

more generally, let T be a general transformation

$$y(n) := T(x(n)) \quad \text{and} \quad x(n) := \sum_{p=-\infty}^{\infty} (x(p)\cdot \delta(n-p))$$

If linear, y1(n) = T(x1(n)), y2(n)=T(x2(n)) => T(a x1(n) + b x2(n)) = a y1(n) + b y2(n)
so, for T() linear:

$$y(n) := \sum_{p=-\infty}^{\infty} T(x(p)\,\delta(n-p)) \quad \text{or we could write as} \quad y(n) := \sum_{p=-\infty}^{\infty} (x(p)\cdot T(\delta(n-p)))$$

and letting h(n,p) = $T(\delta(n-p))$

$$y(n) := \sum_{p=-\infty}^{\infty} (x(p)\cdot h(n,p)) \quad \text{or we could write as} \quad y(n) := \sum_{p=-\infty}^{\infty} \left[x(p)\cdot h_n(p)\right]$$

if time invariant, h(n,p) = h(n-p) simply shifts with time

$$y(n) := \sum_{p=-\infty}^{\infty} (x(p)\cdot h(n-p))$$

1D DFT

The DTFT X(ω) is computed from a discrete-time sampled signal x[n], but has a continuous frequency variable ω. The DFT is sampled in time and frequency, and is equal to the DFT at each frequncy sample, where X[k] = X(ω) at ω = 2πk/N where N is the number of samples in the time and frequency domains. In addition, the time and frequency domains in the case of the DFT both are periodic (hence, circular convolution), with period N.

The DFT and inverse DFT are defined as

$$X(k) := \sum_{n=0}^{N-1} \left[x(n)\, e^{\frac{(-i\cdot 2\cdot \pi)\cdot n\cdot k}{N}} \right]$$

$$x(n) := \frac{1}{N} \cdot \sum_{k=0}^{N-1} \left[X(k)\, e^{\frac{(i\cdot 2\cdot \pi)\cdot n\cdot k}{N}} \right]$$

Circular convolution

$$y(n) := \sum_{p=0}^{N-1} (x(p)\cdot h(\mathrm{mod}(n-p, N)))$$

2D DFT

dummy fnction h(t)

$$f(x,y) := 1$$

assume an NxN pixel image

$$F(u,v) := \frac{1}{N} \cdot \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \left[f(x,y)\, e^{\frac{(-i\cdot 2\cdot \pi)\cdot (u\cdot x + v\cdot y)}{N}} \right]$$

or in separable form:

$$F(u,v) := \frac{1}{N} \cdot \sum_{y=0}^{N-1} \left[e^{\frac{(-i\cdot 2\cdot \pi)\cdot v\cdot y}{N}} \cdot \left[\sum_{x=0}^{N-1} \left[f(x,y)\cdot e^{\frac{(-i\cdot 2\cdot \pi)\cdot u\cdot x}{N}} \right] \right] \right]$$

1-D DFT implemented as a Mathcad program/function

$$\text{dft}(x) := \left| \begin{array}{l} \text{out} \leftarrow x \cdot 0 \\ \text{for } k \in 0 \,..\, \text{rows}(\text{out}) - 1 \\ \quad \text{out}_k \leftarrow \sum_{n=0}^{\text{rows}(x)-1} \left[x_n \cdot e^{\frac{-(i \cdot 2 \cdot \pi \cdot n \cdot k)}{\text{rows}(x)}} \right] \\ \text{out} \end{array} \right.$$

Here we define an algorithm to compute DFT of data stored in a vector x

Example: take the 8-point DFT of a ramp signal x.

$N := 8 \qquad n := 0 \,..\, N - 1 \qquad x_n := n \qquad k := n$

Define the vector x of length N

$X := \text{dft}(x)$

Take the DFT of x

$$x = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix} \qquad X = \begin{pmatrix} 28 \\ -4 + 9.657i \\ -4 + 4i \\ -4 + 1.657i \\ -4 \\ -4 - 1.657i \\ -4 - 4i \\ -4 - 9.657i \end{pmatrix}$$

$$\overrightarrow{|X|} = \begin{pmatrix} 28 \\ 10.453 \\ 5.657 \\ 4.33 \\ 4 \\ 4.33 \\ 5.657 \\ 10.453 \end{pmatrix} \qquad \overrightarrow{\arg(X)} = \begin{pmatrix} 0 \\ 1.963 \\ 2.356 \\ 2.749 \\ -3.142 \\ -2.749 \\ -2.356 \\ -1.963 \end{pmatrix}$$

Find the magnitude and angle of the DFT

Matrix form (Unitary form) of 1-D DFT

Below, simple example to demonstrate mathcad matrix indexing:

$$z_{n,k} := n + k \cdot 10$$

$$z = \begin{pmatrix} 0 & 10 & 20 & 30 & 40 & 50 & 60 & 70 \\ 1 & 11 & 21 & 31 & 41 & 51 & 61 & 71 \\ 2 & 12 & 22 & 32 & 42 & 52 & 62 & 72 \\ 3 & 13 & 23 & 33 & 43 & 53 & 63 & 73 \\ 4 & 14 & 24 & 34 & 44 & 54 & 64 & 74 \\ 5 & 15 & 25 & 35 & 45 & 55 & 65 & 75 \\ 6 & 16 & 26 & 36 & 46 & 56 & 66 & 76 \\ 7 & 17 & 27 & 37 & 47 & 57 & 67 & 77 \end{pmatrix}$$

$N := 8 \qquad n := 0 .. N - 1 \qquad x_n := n \qquad k := n$ Define the vector x of length N

Define the elements of DFT matrix Wdft

$$Wdft_{k,n} := \frac{1}{\sqrt{N}} \cdot e^{\frac{-(i \cdot 2 \cdot \pi \cdot n \cdot k)}{N}}$$

$$Wdft = \begin{pmatrix} 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 \\ 0.354 & 0.25 - 0.25i & -0.354i & -0.25 - 0.25i & -0.354 & -0.25 + 0.25i & 0.354i & 0.25 + 0.25i \\ 0.354 & -0.354i & -0.354 & 0.354i & 0.354 & -0.354i & -0.354 & 0.354i \\ 0.354 & -0.25 - 0.25i & 0.354i & 0.25 - 0.25i & -0.354 & 0.25 + 0.25i & -0.354i & -0.25 + 0.25i \\ 0.354 & -0.354 & 0.354 & -0.354 & 0.354 & -0.354 & 0.354 & -0.354 \\ 0.354 & -0.25 + 0.25i & -0.354i & 0.25 + 0.25i & -0.354 & 0.25 - 0.25i & 0.354i & -0.25 - 0.25i \\ 0.354 & 0.354i & -0.354 & -0.354i & 0.354 & 0.354i & -0.354 & -0.354i \\ 0.354 & 0.25 + 0.25i & 0.354i & -0.25 + 0.25i & -0.354 & -0.25 - 0.25i & -0.354i & 0.25 - 0.25i \end{pmatrix}$$

observe the behavior of the angle of the DFT matrix:

$$\frac{360}{2\cdot\pi}\left(\overrightarrow{\arg(\mathrm{Wdft})}\right) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -45 & -90 & -135 & -180 & 135 & 90 & 45 \\ 0 & -90 & -180 & 90 & 0 & -90 & -180 & 90 \\ 0 & -135 & 90 & -45 & -180 & 45 & -90 & 135 \\ 0 & -180 & 0 & -180 & 0 & -180 & 0 & -180 \\ 0 & 135 & -90 & 45 & -180 & -45 & 90 & -135 \\ 0 & 90 & -180 & -90 & 0 & 90 & -180 & -90 \\ 0 & 45 & 90 & 135 & -180 & -135 & -90 & -45 \end{pmatrix}$$

Note: you may wish to use Format>Result>ComplexThreshold and zeroThreshold
and set to 12 to eliminate small residual roundoff/numerical error in your results display

$$\mathrm{Wdft}\cdot\overline{\left(\mathrm{Wdft}^{T}\right)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Note, the above result implies orthogonal vectors,
i.e., inner product (x,x*)=0
Also, the above shows that DTFT matrix form is unitary.

Unitary matrix means: $A^{-1} = A^{*T} = A^{H}$

Wdtft is unitary, so inverse DCT = conjugate transpose.
Since DTFT is also a symmetric matrix, it so happens that
the conjugate of Wdtft is also the inverse.

So:

$$\mathrm{invWdtft} := \overline{\left(\mathrm{Wdft}^{T}\right)}$$

From the 1D DFT we developed earlier, we had

$$x = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix} \qquad X = \begin{pmatrix} 28 \\ -4 + 9.657i \\ -4 + 4i \\ -4 + 1.657i \\ -4 \\ -4 - 1.657i \\ -4 - 4i \\ -4 - 9.657i \end{pmatrix}$$

Using the matrix form we get the same result:

$$x = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix} \qquad Wdft \cdot x = \begin{pmatrix} 9.899 \\ -1.414 + 3.414i \\ -1.414 + 1.414i \\ -1.414 + 0.586i \\ -1.414 \\ -1.414 - 0.586i \\ -1.414 - 1.414i \\ -1.414 - 3.414i \end{pmatrix}$$

we can check the inverse is conjugate transpose:

$$\left[\left(\overline{Wdft}\right)^{T}\right] \cdot Wdft = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

we can check the inverse DFT:

$$Wdft^{-1} \cdot Wdft \cdot x = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix}$$

1-D DCT (Pratt Eq. 8.3-1)

redefine delta

$$\delta(x) := \delta(x, 0)$$

The 1-D DCT and inverse are

$$F(k) := \sum_{n=0}^{N-1} \left[f(n) \cdot \sqrt{\frac{2 - \delta(k)}{N}} \cos\left[\frac{\pi \cdot k \cdot (n + 0.5)}{N} \right] \right]$$

$$f(n) := \sum_{k=0}^{N-1} \left[F(k) \cdot \sqrt{\frac{2 - \delta(k)}{N}} \cos\left[\frac{\pi \cdot k \cdot (n + 0.5)}{N} \right] \right]$$

2-D DCT (Pratt Eq. 8.3-1)

The 2-D DCT and inverse are defined as

$$F(u,v) := \sum_{x=0}^{N-1} \left[\sum_{y=0}^{N-1} f(x,y) \cdot \sqrt{\frac{2 - \delta(u)}{N}} \cdot \sqrt{\frac{2 - \delta(v)}{N}} \cos\left[\frac{\pi \cdot u \cdot (x + 0.5)}{N} \right] \cdot \cos\left[\frac{\pi \cdot v \cdot (y + 0.5)}{N} \right] \right]$$

$$f(x,y) := \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \left[F(u,v) \cdot \sqrt{\frac{2 - \delta(u)}{N}} \cdot \sqrt{\frac{2 - \delta(v)}{N}} \cdot \cos\left[\frac{\pi \cdot u \cdot (x + 0.5)}{N} \right] \cdot \cos\left[\frac{\pi \cdot v \cdot (y + 0.5)}{N} \right] \right]$$

Matrix form of 1-D DCT

one way to create a DCT matrix is using the if function to load the matrix

$$Wdct_{k,n} := \text{if}\left[k = 0, \sqrt{\frac{1}{N}}\cos\left[\frac{\pi\cdot k\cdot(n+0.5)}{N}\right], \sqrt{\frac{2}{N}}\cos\left[\frac{\pi\cdot k\cdot(n+0.5)}{N}\right]\right]$$

another way to create a DCT matrix is using a short program as below

$$\text{wdctmatrix}(nn) := \left|\begin{array}{l} \text{for } k \in 0..nn-1 \\ \quad \text{for } n \in 0..nn-1 \\ \quad\quad w_{k,n} \leftarrow \sqrt{\frac{2}{nn}}\cos\left[\frac{\pi\cdot k\cdot(n+0.5)}{nn}\right] \\ \quad\quad w_{k,n} \leftarrow \sqrt{\frac{1}{nn}}\cos\left[\frac{\pi\cdot k\cdot(n+0.5)}{nn}\right] \text{ if } k = 0 \\ w \end{array}\right.$$

$Wdct := \text{wdctmatrix}(N)$

$$Wdct = \begin{pmatrix} 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 \\ 0.49 & 0.416 & 0.278 & 0.098 & -0.098 & -0.278 & -0.416 & -0.49 \\ 0.462 & 0.191 & -0.191 & -0.462 & -0.462 & -0.191 & 0.191 & 0.462 \\ 0.416 & -0.098 & -0.49 & -0.278 & 0.278 & 0.49 & 0.098 & -0.416 \\ 0.354 & -0.354 & -0.354 & 0.354 & 0.354 & -0.354 & -0.354 & 0.354 \\ 0.278 & -0.49 & 0.098 & 0.416 & -0.416 & -0.098 & 0.49 & -0.278 \\ 0.191 & -0.462 & 0.462 & -0.191 & -0.191 & 0.462 & -0.462 & 0.191 \\ 0.098 & -0.278 & 0.416 & -0.49 & 0.49 & -0.416 & 0.278 & -0.098 \end{pmatrix}$$

$$Wdct\cdot\left(\overline{Wdct}\right)^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Note, orthogonal vectors!
Is unitary and orthogonal (since real).

Unitary: $A^{-1} = A^{*T} = A^H$
Orthogonal: $A^{-1} = A^T$

Is orthogonal transform, so inverse DCT = transpose

Note: Wdct is full rank, invertible, full/complete basis for vector space

Comparison of DTFT and DCT basis functions:

Dtft basis functions

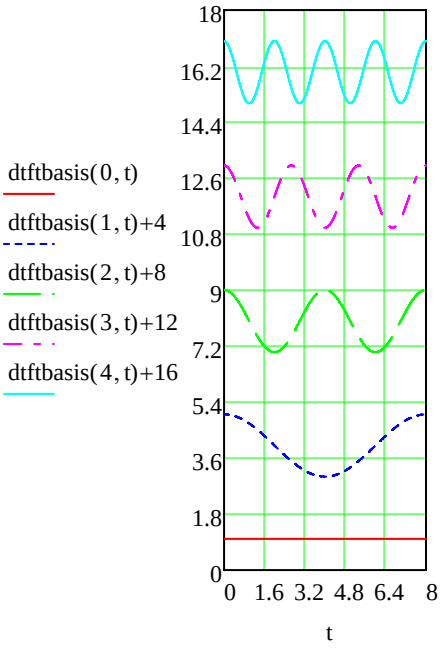
$N := 8$

$$dtftbasis(u,t) := Re\left[e^{\frac{-(i\cdot 2\cdot\pi\cdot u\cdot t)}{N}}\right]$$

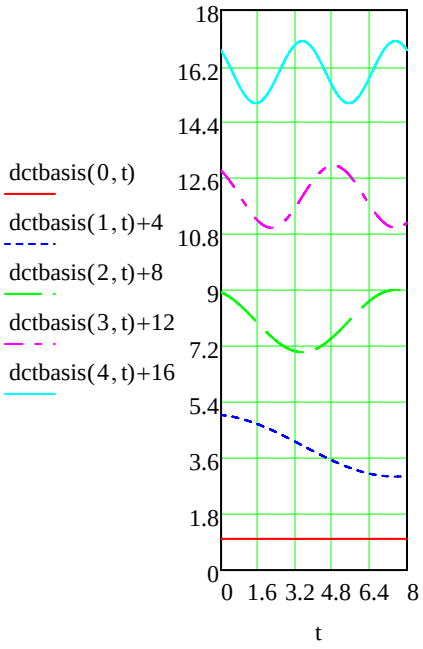
Dct basis functions

$$dctbasis(u,t) := \cos\left[\frac{\pi\cdot u\cdot(t+0.5)}{N}\right]$$

DFT basis functions $N := 8$



DCT basis functions $N := 8$



Q1 : Change the above 2 plots to compare the first 8 basis functions, instead of the first 5

Matrix form of 1-D Hadamard Transform

$$\text{Whad}(\exp) := \left| \begin{array}{l} H1 \leftarrow \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \\ \text{for } n \in 1 .. \exp - 1 \\ \quad \left| \begin{array}{l} Ht \leftarrow \text{augment}(H1, H1) \\ Hb \leftarrow \text{augment}(H1, -H1) \\ H \leftarrow \text{stack}(Ht, Hb) \\ H1 \leftarrow H \cdot \frac{1}{\sqrt{2}} \end{array} \right. \\ H \leftarrow H1 \\ H \end{array} \right.$$

Hadamard matrix size will be 2^{exp}

$$\text{Whad} := \text{Whad}(\log(N, 2))$$

$$\text{Whad} = \begin{pmatrix} 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 \\ 0.354 & -0.354 & 0.354 & -0.354 & 0.354 & -0.354 & 0.354 & -0.354 \\ 0.354 & 0.354 & -0.354 & -0.354 & 0.354 & 0.354 & -0.354 & -0.354 \\ 0.354 & -0.354 & -0.354 & 0.354 & 0.354 & -0.354 & -0.354 & 0.354 \\ 0.354 & 0.354 & 0.354 & 0.354 & -0.354 & -0.354 & -0.354 & -0.354 \\ 0.354 & -0.354 & 0.354 & -0.354 & -0.354 & 0.354 & -0.354 & 0.354 \\ 0.354 & 0.354 & -0.354 & -0.354 & -0.354 & -0.354 & 0.354 & 0.354 \\ 0.354 & -0.354 & -0.354 & 0.354 & -0.354 & 0.354 & 0.354 & -0.354 \end{pmatrix}$$

$$\text{Whad} \cdot \text{Whad}^{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Note, orthogonal vectors!
Is unitary and orthogonal (since real).

Unitary: $A^{-1} = A^{*T} = A^{H}$

Orthogonal: $A^{-1} = A^{T}$

Is orthogonal transform, so inverse DCT = transpose

Sequency is the number of zero-crossings in each row,
we can write a function to compute the sequency of each row

$$\text{sequency}(x) := \left| \begin{array}{l} \text{for } r \in 0 .. \text{rows}(x) - 1 \\ \quad ss_r \leftarrow 0 \\ \text{for } r \in 0 .. \text{rows}(x) - 1 \\ \quad \text{for } c \in 0 .. \text{cols}(x) - 2 \\ \quad\quad ss_r \leftarrow ss_r + 1 \text{ if } \frac{x_{r,c}}{x_{r,c+1}} \le 0 \\ ss \end{array} \right.$$

Note that the sequency is out of order:

$$\text{seqW} := \text{sequency}(\text{Whad}) \qquad \text{seqW} = \begin{pmatrix} 0 \\ 7 \\ 3 \\ 4 \\ 1 \\ 6 \\ 2 \\ 5 \end{pmatrix}$$

Matrix form 1-D Sequency-ordered Hadamard Transform

$N = 8$

$$
\begin{aligned}
&\text{seqHad}(\text{exp}) := \\
&\quad H1 \leftarrow \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \\
&\quad \text{for } n \in 1 .. \text{exp} - 1 \\
&\quad\quad Ht \leftarrow \text{augment}(H1, H1) \\
&\quad\quad Hb \leftarrow \text{augment}(H1, -H1) \\
&\quad\quad H \leftarrow \text{stack}(Ht, Hb) \\
&\quad\quad H1 \leftarrow H \cdot \frac{1}{\sqrt{2}} \\
&\quad H \leftarrow H1 \\
&\quad ww \leftarrow H \\
&\quad seqW \leftarrow \text{sequency}(ww) \\
&\quad out \leftarrow ww \\
&\quad \text{for } n1 \in 0 .. \text{rows}(seqW) - 2 \\
&\quad\quad \text{for } n2 \in n1 + 1 .. \text{rows}(seqW) - 1 \\
&\quad\quad\quad se1 \leftarrow seqW_{n1} \\
&\quad\quad\quad se2 \leftarrow seqW_{n2} \\
&\quad\quad\quad rn1 \leftarrow \left(out^{T}\right)^{\langle n1 \rangle} \quad \text{if } se2 < se1 \\
&\quad\quad\quad rn2 \leftarrow \left(out^{T}\right)^{\langle n2 \rangle} \quad \text{if } se2 < se1 \\
&\quad\quad\quad seqW_{n2} \leftarrow se1 \quad \text{if } se2 < se1 \\
&\quad\quad\quad seqW_{n1} \leftarrow se2 \quad \text{if } se2 < se1 \\
&\quad\quad\quad \text{for } n3 \in 0 .. \text{rows}(seqW) - 1 \quad \text{if } se2 < se1 \\
&\quad\quad\quad\quad out_{n1, n3} \leftarrow rn2_{n3} \\
&\quad\quad\quad\quad out_{n2, n3} \leftarrow rn1_{n3} \\
&\quad\quad\quad se1 \leftarrow se2 \\
&\quad out
\end{aligned}
$$

$$Wseqhad := seqHad(\log(N, 2))$$

$$Wseqhad = \begin{pmatrix} 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 \\ 0.354 & 0.354 & 0.354 & 0.354 & -0.354 & -0.354 & -0.354 & -0.354 \\ 0.354 & 0.354 & -0.354 & -0.354 & -0.354 & -0.354 & 0.354 & 0.354 \\ 0.354 & 0.354 & -0.354 & -0.354 & 0.354 & 0.354 & -0.354 & -0.354 \\ 0.354 & -0.354 & -0.354 & 0.354 & 0.354 & -0.354 & -0.354 & 0.354 \\ 0.354 & -0.354 & -0.354 & 0.354 & -0.354 & 0.354 & 0.354 & -0.354 \\ 0.354 & -0.354 & 0.354 & -0.354 & -0.354 & 0.354 & -0.354 & 0.354 \\ 0.354 & -0.354 & 0.354 & -0.354 & 0.354 & -0.354 & 0.354 & -0.354 \end{pmatrix}$$

The new method produces a Hadamard with ordered sequency

$$Wseqhad \cdot Wseqhad^{T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad sequency(Wseqhad) = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix}$$

Note: Hadamard is full rank, invertible, full/complete basis for vector space

Q2 : For N=8 as above, show that inverse Hadamard = itself = Wseqhad:

Comparison of DTFT and Sequency-ordered Hadamard basis functions:

Dtft basis functions

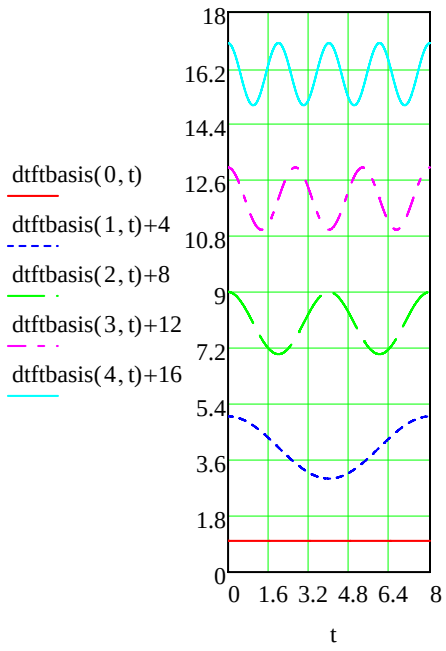
Hadamard basis functions

$\text{rect}(t) := \Phi(t + 0.5) - \Phi(t - 0.5)$

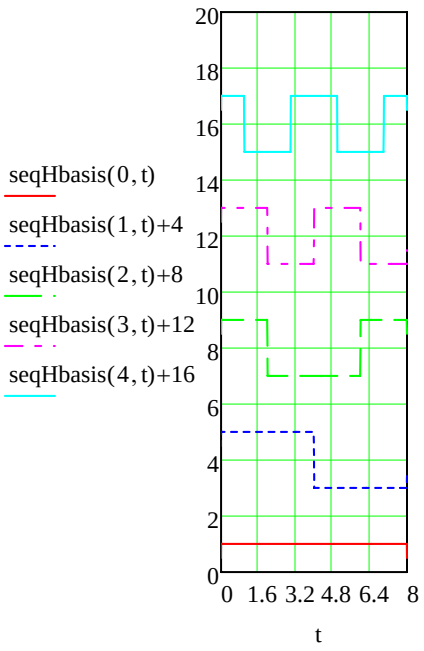
$N = 8$

$$\text{dtftbasis}(u,t) := \text{Re}\left[e^{\frac{-(i\cdot 2\cdot\pi\cdot u\cdot t)}{N}}\right] \qquad \text{seqHbasis}(u,t) := \sqrt{N}\cdot\sum_{n=0}^{N-1}\left[\left[\left(\text{Wseqhad}^{T}\right)^{\langle u\rangle}\right]_{n}\cdot\text{rect}\left(t-n-\frac{1}{2}\right)\right]$$

DFT basis functions $N := 8$



Hadamard basis funct $N := 8$



Q3 : Change the above 2 plots to compare the first 8 basis functions, instead of the first 5.

Comparison of DTFT, DCT, and Sequency-ordered Hadamard transforms of a ramp signal

$N := 16 \quad n := 0 .. N - 1 \quad x_n := n \quad k := n$ Define the vector x of length N

Q4,Q5,Q6 : For the N=16 point ramp signal x shown below, compute the 1-D DFT,DCT, and Hadamard transforms. Use above functions to compute the transfofm matrices M, then write Mx= to show the result.

DTFT result: **DCT result:** **Sequency-ordered Hadamard result:**

$x =$

	0
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15

2-D Transforms

The 2-D DFT, DCT, and Hadamard can all be generated from the 1-D matrix forms of each transform as follows. All three transforms are unitary and separable, and so the same procedures will apply.

Unitary Transforms

- For column vector $\underline{x}$ and $\underline{y}$, and transformation matrix A

  $\underline{\mathbf{y}} = \mathbf{A}\,\underline{\mathbf{x}}$

- Matrix **A** is unitary if (Pratt Eq. 8.1-10)

  $\mathbf{A}\,\mathbf{A}^{H} = \mathbf{I}$ or $\mathbf{A}^{H} = \mathbf{A}^{-1}$

- Fourier matrix **W** is unitary, so 1D DFT is unitary

- Since 2D DFT is separable, (Pratt Eq. 6.2-2, 6.2-8)

  $\underline{\mathbf{F}} = \mathbf{T}\,\underline{\mathbf{f}} = \mathbf{T}_c \otimes \mathbf{T}_r\,\underline{\mathbf{f}} = \mathbf{W} \otimes \mathbf{W}\,\underline{\mathbf{f}}$ **with** $\underline{\mathbf{F}}$ and $\underline{\mathbf{f}}$ vectors

- **And in matrix form** (see Pratt Eq. 6.2-10, 8.1-13)

  $\mathbf{F} = \mathbf{W}\,\mathbf{f}\,\mathbf{W}^{T}$ with **F** and **f** matrices

  And with inverse:

  $\mathbf{f} = \mathbf{W}^{-1}\,\mathbf{F}\,(\mathbf{W}^{T})^{-1} = \mathbf{W}^{-1}\,\mathbf{F}\,(\mathbf{W}^{-1})^{T} = \mathbf{W}^{H}\,\mathbf{F}\,(\mathbf{W}^{H})^{T}$

A simple initial test image for transforms

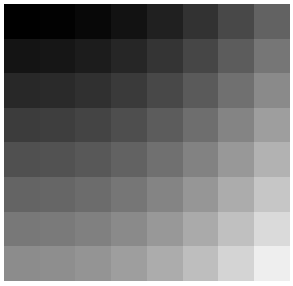
$x := 0$ Sometimes it is best to clear a variable before re-use

$N := 8$ $n := 0 .. N - 1$ $x_n := n$ $k := n$ Define the vector x of length N

First, create a 2-D image matrix for test purposes

$r := n$ $c := n$

$x_{r,c} := 2 \cdot \left(10 \cdot r + c^2\right)$



x

$$x = \begin{pmatrix} 0 & 2 & 8 & 18 & 32 & 50 & 72 & 98 \\ 20 & 22 & 28 & 38 & 52 & 70 & 92 & 118 \\ 40 & 42 & 48 & 58 & 72 & 90 & 112 & 138 \\ 60 & 62 & 68 & 78 & 92 & 110 & 132 & 158 \\ 80 & 82 & 88 & 98 & 112 & 130 & 152 & 178 \\ 100 & 102 & 108 & 118 & 132 & 150 & 172 & 198 \\ 120 & 122 & 128 & 138 & 152 & 170 & 192 & 218 \\ 140 & 142 & 148 & 158 & 172 & 190 & 212 & 238 \end{pmatrix}$$

2-D DFT using the cfft fast Fourier transform

This Mathcad built-in function cfft uses a 2-D FFT algorithm

$Xdtft := cfft(x)$

$magXdtft := \overrightarrow{|Xdtft|}$

$$magXdtft = \begin{pmatrix} 840 & 154.794 & 80 & 60.72 & 56 & 60.72 & 80 & 154.794 \\ 209.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 113.137 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 86.591 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 86.591 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 113.137 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 209.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

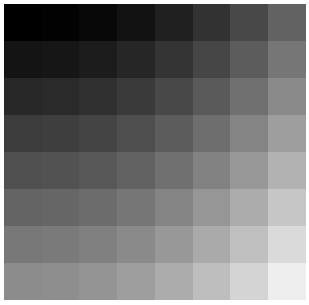
2-D DTFT using matrices (Pratt Eq. 8.1-13a, 8.2-15a)

$$Xdft := Wdft \cdot x \cdot (Wdft)^T$$

Note: this would NOT be a fast algorithm, since it is implemented as a regular matrix multiply

$$magXdft := \overrightarrow{|Xdft|}$$

$$magXdft = \begin{pmatrix} 840 & 154.794 & 80 & 60.72 & 56 & 60.72 & 80 & 154.794 \\ 209.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 113.137 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 86.591 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 86.591 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 113.137 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 209.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



x

input image x

$$250 \cdot \frac{magXdtft}{\max(magXdtft)}$$

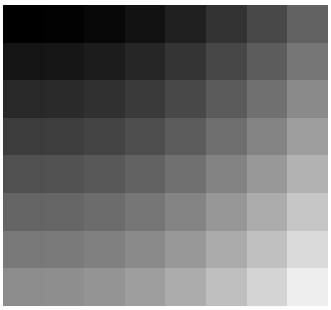
Magnitude spectrum image

2-D inverse DFT using matrices ((Pratt Eq. 8.3-1)

$$xx := \left(\overline{Wdft}\right)^{T} \cdot Xdft \cdot \overline{Wdft}$$

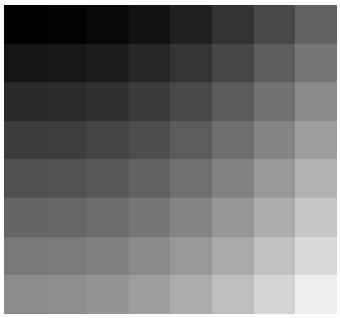
Check that we can recover the original image using the inverse DFT

$$xx = \begin{pmatrix} 0 & 2 & 8 & 18 & 32 & 50 & 72 & 98 \\ 20 & 22 & 28 & 38 & 52 & 70 & 92 & 118 \\ 40 & 42 & 48 & 58 & 72 & 90 & 112 & 138 \\ 60 & 62 & 68 & 78 & 92 & 110 & 132 & 158 \\ 80 & 82 & 88 & 98 & 112 & 130 & 152 & 178 \\ 100 & 102 & 108 & 118 & 132 & 150 & 172 & 198 \\ 120 & 122 & 128 & 138 & 152 & 170 & 192 & 218 \\ 140 & 142 & 148 & 158 & 172 & 190 & 212 & 238 \end{pmatrix}$$



x

original image x



xx

IDFT

$xx := xx - xx$ Sometimes it is best to clear a variable before re-use

2-D DCT using matrices (Pratt Eq. 8.3-1)

Q7 : : For the same image x in the DFT above, compute the 2-D DCT transform below using a matrix method similar to the DFT form Xdft=Wdft x Wdft ^{T}.

Xdct := fix this!

magXdct :=

magXdct =

DFT

$$250 \cdot \frac{magXdtft}{max(magXdtft)}$$

DCT

$$250 \cdot \frac{magXdct}{max(magXdct)}$$

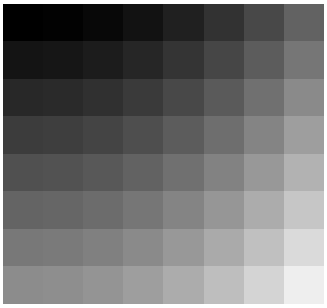
2-D inverse DCT using matrices (Pratt Eq. 8.3-1)

Q8 : For the same image x in the DFT above, compute the inverse 2-D DCT transform below, and show that the original image is recovered.

xx :=

$$xx = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

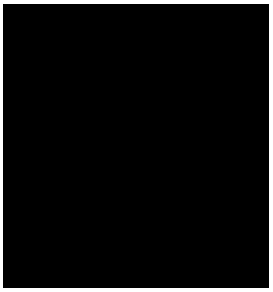
x



x

original image x

IDCT



xx

IDCT

xx := xx − xx

Sometimes it is best to clear a variable before re-use

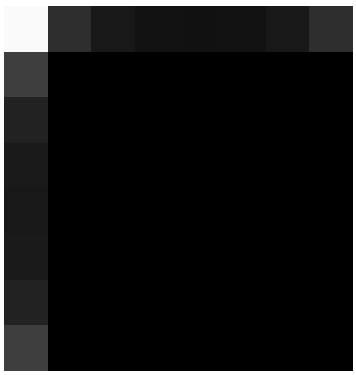
2-D Sequency-ordered Hadamard using matrices

Q9 : For the same image x in the DFT above, compute the 2-D Hadamard transform below.

Xseqhad :=

magXseqhad :=

magXseqhad =



$$250 \cdot \frac{magXseqhad}{max(magXseqhad)}$$

$$250 \cdot \frac{magXdtft}{max(magXdtft)}$$

DFT

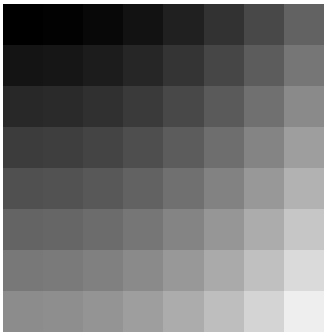
Hadamard

2-D inverse Hadamard using matrices

Q10 : For the same image x in the Hadamard above, compute the inverse 2-D Hadamard transform below, and show that the original image is recovered.

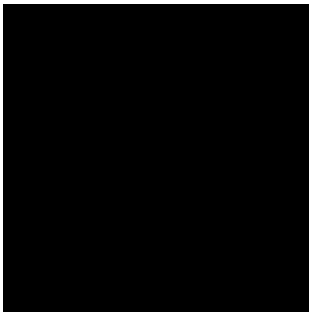
xx :=

$$xx = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



x

original image x



xx

inverse Hadamard

Q11,Q12,Q13 : For the 256x256 pixel wheel.gif image, compute the 2-D DFT,DCT, and sequency-ordered Hadamard transforms and display the results as spectral images. Use the following logarithmic plotting function for your output images. You mustremove the DC component from the image x before taking transforms (see last line below).

$$magXdct := \overrightarrow{|Xdct|}$$

$$maxdct := \max(magXdct)$$

$$dctplot := 3 \cdot \overrightarrow{\left(20 \cdot \log\left(1 + \frac{10000 \cdot magXdct}{maxdct}\right)\right)}$$

$\max(dctplot) =$

Logarithmic plotting function for output spectra

$infile := READBMP("wheel.gif")$

$x := infile$

$rows(x) =$ $cols(x) =$ $x_{1,1} =$

infile

$x := infile - mean(infile)$

Note: you cannot plot x anymore, because it has negative values